

A note on esApply

`exprSets` are complex objects. We will think of them as linked arrays: the `exprs` element of an *exprSet* is $G \times N$, where G is the number of genes on a chip and N is the number of tissues analyzed, and the `pData` element of the associated `phenoData` element is $N \times p$, where p is the number of phenotypic or demographic, etc., variables collected.

Abstractly, we are often interested in evaluating functions $f(y; x)$ where y is an N -vector of expression results for a specific gene and x is an N -dimensional structure, coordinated with y , that distinguishes elements of y for processing in the function f . A basic problem is to guarantee that the j th element of y is correctly associated with the j th component of x .

As an example, let's consider `eset` which is an *exprSet* supplied with Biobase. We will print a little report, then the first N -vector of gene expressions and some covariate data:

```
> print(eset)
```

```
Expression Set (exprSet) with
```

```
500 genes
```

```
26 samples
```

```
      phenoData object with 3 variables and 26 cases
```

```
varLabels
```

```
cov1: Covariate 1; 2 levels
```

```
cov2: Covariate 2; 2 levels
```

```
cov3: Covariate 3; 3 levels
```

```
> print(exprs(eset)[1, ])
```

	A	B	C	D	E	F	G	H
	192.7420	85.7533	176.7570	135.5750	64.4939	76.3569	160.5050	65.9631
	I	J	K	L	M	N	O	P
	56.9039	135.6080	63.4432	78.2126	83.0943	89.3372	91.0615	95.9377
	Q	R	S	T	U	V	W	X
	179.8450	152.4670	180.8340	85.4146	157.9890	146.8000	93.8829	103.8550
	Y	Z						
	64.4340	175.6150						

```
> print(pData(eset)[1:2, 1:3])
```

	cov1	cov2	cov3
A	1	1	1
B	1	1	1

Now let's see how expressions and a covariate are related:

```
> print(rbind(exprs(eset[1, ]), cov1 = t(pData(eset))[1, ]))
```

	A	B	C	D	E	F	G	H
AFFX-MurIL2_at	192.742	85.7533	176.757	135.575	64.4939	76.3569	160.505	65.9631
cov1	1.000	1.0000	1.000	1.000	1.0000	1.0000	1.000	1.0000
	I	J	K	L	M	N	O	P
AFFX-MurIL2_at	56.9039	135.608	63.4432	78.2126	83.0943	89.3372	91.0615	95.9377
cov1	1.0000	1.000	1.0000	1.0000	1.0000	2.0000	2.0000	2.0000
	Q	R	S	T	U	V	W	X
AFFX-MurIL2_at	179.845	152.467	180.834	85.4146	157.989	146.8	93.8829	103.855
cov1	2.000	2.000	2.000	2.0000	2.000	2.0	2.0000	2.000
	Y	Z						
AFFX-MurIL2_at	64.434	175.615						
cov1	2.000	2.000						

A function that evaluates the difference in median expression across strata defined using an abstract covariate *x* is

```
> medContr <- function(y, x) {
+   ys <- split(y, x)
+   median(ys[[1]]) - median(ys[[2]])
+ }
```

We can apply this to a small *exprSet* that gives back the data listed above:

```
> print(apply(exprs(eset[1, , drop = F]), 1, medContr, pData(eset)[["cov1"]]))
```

AFFX-MurIL2_at
-20.7607

That's a bit clumsy. This is where **esApply** comes in. We pay for some simplicity by following a strict protocol for the definition of the statistical function to be applied.

```
> medContr1 <- function(y) {
+   ys <- split(y, cov1)
+   median(ys[[1]]) - median(ys[[2]])
+ }
> print(esApply(eset, 1, medContr1)[1])
```

AFFX-MurIL2_at
-20.7607

The manual page on **esApply** has a number of additional examples that show how applicable functions can be constructed and used. The important thing to note is that the applicable functions *know* the names of the covariates in the **pData** dataframe.

This is achieved by having an environment populated with all the variables in the **phenoData** component of the *exprSet* put in as the environment of the function that will be applied. If that function already has an environment we retain that but in the second position. Thus, there is some potential for variable shadowing.